Appendix B

Diagnostic Messages

B.1 Diagnostic Messages from the vcc Command

This section lists the diagnostic messages that can be generated by the **vcc** command program. For each message, the description gives the message text, an explanation of the message, and suggested actions to correct the error.

error: no optimization allowed with debug

Warning: You invoked the **vcc** command with both the **–O** option and the **–g** option. VAX C provides debugging, but not optimization and debugging at the same time.

User Action: In the future, invoke the debugger without the **–O** option to avoid this error.

error: unable to execute the assembler

Fatal: Either the as assembler is not available on the system or there is a protection violation that prevents its use.

User Action: Check that the as assembler is available and that its protection is set to r-x.

error: unable to execute the compiler

Fatal: Either the VAX C compiler is not available on the system or there is a protection violation that prevents its use.

User Action: Check that the VAX C compiler is available and that its protection is set to r-x.

error: unable to execute the linker

Fatal: Either the linker is not available on the system or there is a protection violation that prevents its use.

User Action: Check that the linker is available and that its protection is set to r-x.

error: unable to execute the om

Fatal: Either the om utility is not available on the system or there is a protection violation that prevents its use.

User Action: Check that the om utility is available and that its protection is set to r-x.

error: unable to execute the preprocessor

Fatal: Either the c preprocessor (cpp) is not available on the system or there is a protection violation that prevents its use.

User Action: Check that cpp is available and that its protection is set to r-x.

error: will overwrite ****

Fatal: The listing file must not have a .c extension. Execution is halted so that the specified file is not overwritten.

User Action: Change the listing file name so that the extension is not .c.

B.2 Diagnostic Messages from the VAX C Compiler

This section lists the VAX C compiler diagnostic messages. The format of the error messages is as follows:

"file-name", line nnn: %severity-mnemonic, msg

The file-name is replaced by the name of the source file that generated the message, and nnn is replaced by the line number that identifies the location in the source file where the error was detected. The error severity code is followed by a hyphen that is followed by a brief mnemonic message abbreviation that provides a key to the alphabetized list that appears later in this section. The last part of the message, msg, is replaced by the message text that is associated with the mnemonic.

The severity can be one of the following single-letter codes representing the meanings shown:

F Fatal
E Error
W Warning
I Informational
S Success

For each message, the descriptions that follow give the mnemonic, the message text, an explanation of the message, and suggested actions to correct the error.

Some messages substitute information from the program in the message text. In this appendix, the portion of the text to be substituted is shown as "****" or ****. If quotes appear around the asterisks, quotes appear in the substituted message.

You can suppress the warning and informational messages with the **[no]warnings** option on the **vcc** command line. You may want to do this so that the compiler broadcasts only the most severe messages to the terminal. For more information about the [NO]WARNINGS option, see Chapter 2.

ANACHRONISM, The "****" operator is an obsolete form, and may not be portable.

Informational: You used an old-style assignment operator such as =+ or =*.

User Action: For the program to be portable, reverse the order of the operator parts. For example, change =+ to += and change =* to *=. VAX C still supports the old-style operators, but they may not be supported

by other C compilers, and they are not guaranteed to be supported in future releases of VAX C.

ARGINVSTRPTR, The *** argument of "***" built-in function is not a pointer to structure or union with size: 1, 2, or 4 bytes.

Error: A built-in function that takes a **struct** argument was not passed a **struct** of the appropriate size.

User Action: Correct the call to the built-in function to pass the correct number and type of arguments.

ARGLISTOOLONG, Function reference specifies an argument list whose length exceeds the VAX architecture limit.

Error: The size of your argument list in the function call exceeded 255 longwords.

User Action: Rewrite the function definition and function call with a list whose member(s) take less space; for example, by passing floating-point and structure arguments by reference rather than by value. Recall that floating-point arguments occupy two longwords, and that structures passed by value occupy as many longwords as are necessary to contain the whole structure.

ARGNOFLOAT, The *** argument of "***" builtin function may not be floating point. The argument has been converted to an integer.

Warning: An argument to a built-in function has a floating-point type when it should have an integer type.

User Action: Correct the call to the built-in function to pass the correct number and type of arguments. If you wish to pass a **float** argument, use an explicit cast.

ARGNOTINTPTR, The *** argument of "***" builtin function is not a pointer to integer.

Error: An argument to a built-in function does not have the required type of pointer to some type of integer.

User Action: Correct the call to the built-in function to pass the correct number and type of arguments. Check the arguments for missing address-of operators (&).

ARGNOTLVALUE, The *** argument of "***" builtin function is not an lvalue.

Error: An argument that is required to be an lvalue is a non-lvalue expression.

User Action: Correct the call to the built-in function to pass the correct number and type of arguments. Make sure the appropriate arguments are lvalues.

ARGNOTPTRVAL, The *** argument of "***" builtin function is not a pointer.

Error: An argument that is required to be some type of pointer does not have a pointer type.

User Action: Correct the call to the built-in function to pass the correct number and type of arguments. Check the arguments for missing "address of" operators (&).

ARGOVERFLOW, Length of the argument list for macro "****" exceeds buffer capacity; overflowing argument(s) considered to be null.

Warning: The total length of the arguments in a macro reference exceeded the compiler's capacity to store the arguments prior to substitution.

User Action: Shorten or eliminate one or more arguments.

ARGREADONLY, The *** argument of "*** builtin function is read-only.

Error: An argument that is used by the function to modify memory is a pointer to const or read-only memory.

User Action: Correct the call to the built-in function to pass the correct number and type of arguments. Make sure that arguments that the function uses to change memory point to writeable memory.

ARGSTOOFEW, Argument list for builtin function "***" contains too few arguments; the builtin function is being ignored.

Error: Not all of the required arguments were specified.

User Action: Correct the call to the built-in function to pass the correct number and type of arguments.

ARGSTOOMANY, Argument list for builtin function "***" contains too many arguments; excess arguments ignored.

Warning: A function was called with extra arguments.

User Action: Correct the call to the built-in function to pass the correct number and type of arguments.

BADCODE, Invalid code generation sequence.

Fatal: An internal compiler error occurred.

User Action: Gather as much information as you can about the conditions in effect when the error occurred, and submit an SPR.

BADPSECT, The program section (psect) specified by this statement has conflicting 'nowrite' attributes with another definition of the same program section.

Warning: You specified two or more references to the same program section, and the attributes of the references do not correspond.

For example, this message appears when two **globaldef** definitions exist for the same name, but only one specifies the **readonly** storage class.

User Action: Make all references to a program section consistent.

BUGCHECK, Compiler bug check during ****. Submit an SPR with a problem description.

Fatal: An internal error occurred during the specified phase of compilation.

User Action: Gather as much information as possible about the conditions under which the error occurred, including the phase of compilation, and submit an SPR (see the *Basic Installation Guide*).

BUILTARGCONV, The *** argument of "***" builtin function has been converted from pointer to arithmetic type.

Warning: An argument that should have an integer or floating-point type had a pointer type.

User Action: Correct the call to the built-in function to pass the correct number and type of arguments. If you want to pass a pointer argument to an arithmetic argument, use an explicit cast.

CANTINLINECALL, Can't inline this call to "****" as requested because not enough actual parameters are supplied in the call.

Informational: The number of parameters supplied in a call to the function is fewer than the number of formal parameters declared and used in the function. Function calls that do not supply enough parameters will not be expanded inline.

User Action: Change the call so that all necessary parameters are supplied, or eliminate unneeded formal parameters from the function.

CANTINLINECALL, Can't inline this call to "****" as requested because an offset into a by value parameter exceeds size of actual.

Informational: The actual value of a parameter provided in a call was smaller in size than the corresponding formal parameter of the function. Use of the formal parameter requires the full amount of storage. This indicates that the type of the formal parameter does not match the type of the actual value provided in the call.

User Action: Change either the formal parameter or the actual value provided in the call so that the type of the formal parameter matches the type of the actual value.

CANTINLINEPROC, Can't inline "****" as requested because a variable offset into a by value parameter is used.

Informational: A formal parameter is referenced with a run-time variable subscript. This is usually a parameter of type **struct** containing a field that is an array. Functions that use formal parameters in this way will not be expanded inline.

User Action: Pass a pointer to the **struct** instead of the **struct** itself, or remove the pragma that requests that the function be expanded inline.

CANTINLINEPROC, Can't inline "****" as requested because it declares an exception handler.

Informational: It was requested that a function be expanded inline. However, that function declares an exception handler. Since the function would not have a call frame, it cannot have an exception handler if it is to be expanded inline.

User Action: Eliminate the exception handler, or remove the pragma that requests that the function be expanded inline.

CANTINLINEPROC, Can't inline "****" as requested because it takes the address of a passed by value parameter.

Informational: The function uses operators such as & to take the address of a formal parameter, or uses the *varargs* package. These practices prevent inline expansion of the function because it may store parameters in registers (which have no address) after inline expansion, and because you may have been relying upon the parameters being adjacent to each other in memory, which will not be true after inline expansion.

User Action: If possible, code the function without using the address of the parameter, or if an address is needed, then change the formal parameter to be a pointer to the value. If the *varargs* package is used, then remove the pragma requesting that the function be expanded inline.

CASECONSTANT, Case label value is not a constant expression.

Error: You specified a value in a **case** label that was not a constant.

User Action: Replace the **case** value with a valid constant expression.

CMPLXINIT, "****" is too complex to initialize.

Warning: The depth of the indicated aggregate variable exceeded the limit of 32 levels.

User Action: Simplify or correct the initializer list or declaration, or initialize the variable within an assignment statement.

COMPILERR, Previous errors prevent continued compilation. Please correct reported errors and recompile.

Fatal: The compiler detected too many errors to continue.

User Action: Correct the errors reported in the previous compiler messages.

CONBUILTARG, Constant expression required for "****" argument of "****" builtin function.

Error: Some built-in functions require that certain arguments be constants or expressions that the compiler can evaluate at compile time to produce a constant. If a nonconstant expression is used for any such argument, this error message is issued.

User Action: Replace the offending argument expression with a constant. If the structure of the program requires that the built-in function be called with different values that can only be calculated at run time, consider using a **switch** statement to call the built-in function with different (constant) arguments on the basis of the run-time expression.

CONFLICTDECL, This declaration of "****" conflicts with a previous declaration of the same name.

Warning: The compiler determined that both declarations see the same object, yet the two declarations conflict in data-type or storage-class organization.

In addition, for external variables and global symbols, the compiler may detect conflicting storage-class specifiers. If the compiler issues an error message for this reason, the program may be correct; issuing a message in this instance is a warning against possible programming errors.

User Action: If the declarations see the same object, make sure that they specify the same types and organizations. Otherwise, either rename one of the identifiers or separate the scopes of the declarations.

DEFTOOLONG, Text in #define preprocessor directive is too long; directive ignored.

Warning: The length of the token string in the **#define** directive exceeded the implementation's limit.

User Action: Simplify the directive.

DIVIDEZERO, Constant expression includes divide by zero; the result has been replaced with 0.

Warning: A division by 0 was encountered in a constant expression. The expression was replaced by 0.

User Action: Make sure that no divisors in the expression can evaluate to 0.

DUPCASE, Duplicate case label value "****".

Error: You specified more than one **case** for the indicated value in a **switch** statement. (The cases must be unique.)

User Action: Change the **case** labels and combine the cases, or both, as appropriate.

DUPDEFAULT, Duplicate default label.

Error: You specified more than one default case in the same **switch** statement.

User Action: Combine the cases or make other changes necessary to eliminate the duplicate(s).

DUPDEFINITION, Duplicate definition of "****".

Warning: The named definition appeared more than once in the program.

The two definitions are essentially the same. Both definitions specify the same data types and organizations, but there may be differences in the values, initializers, or array bounds. If the name is a function, there may be a difference in the number or types of parameters, or in the contents of the function body.

User Action: The purpose of this message is to call a possible programming error to your attention.

DUPINLINEFUNC, Duplicate [no]inline function "****".

Warning: You duplicated a function name in one or more pragma declarations.

User Action: Change the name of the function declaration.

DUPLABEL, Duplicate label "****".

Error: You specified duplicates of the indicated label in the same function. (Label identifiers must be unique within a function definition.)

User Action: Rewrite the labels (and the **goto** statements that see them) to eliminate the duplicates.

DUPLISTITEM, Duplicate list item "****" ignored.

Warning: You specified the same name more than once in a list of arguments to a **#pragma** directive. For example, in the following **#pragma**, the second appearance of variable a is redundant and is ignored:

```
#pragma noinline (a, b, a)
```

Similarly, the second occurrence of variable y in the following example is redundant, as argument lists for some **#pragma** directives are concatenated:

```
#pragma noinline (x, y)
    .
    .
    .
#pragma noinline (y, z)
```

User Action: Remove the duplicate argument if it is redundant; otherwise, check for misspellings.

DUPMAINFUNC, Duplicate main function.

Warning: You defined two or more main functions in a single compilation unit.

A main function is either a function with the name main or a function with the main_program option. If the compilation unit contains more than one main function, the compiler recognizes only the first as the main function.

User Action: Make sure that there is only one main function defined in the compilation unit.

DUPMEMBER, Duplicate declaration of member "****".

Warning: You declared two members with the same name in the same structure.

User Action: Rename one of the members or remove one of the member declarations.

DUPPARAMETER, Duplicate parameter "****" ignored.

Warning: The stated function parameter occurred more than once in the function's formal parameter list. For example:

```
funct(a,b,c,a) { }
```

All occurrences of the parameter after the first are ignored.

User Action: Remove or change the duplicate parameter identifier.

ENUMCLASH, Mismatched enum type in "****" operation.

Warning: The indicated operation combined an **enum** variable or value with a value that has a nonmatching type. The compiler issues this message if you specify the **–V standard=portable** option on the **vcc** command line.

User Action: Use a cast operation to cast either the **enum** value or the other value to a matching type.

ENUMOP, "****" is an undefined operation for enum values; enum operand(s) converted to int.

Warning: You used an **enum** variable or constant with an arithmetic or bitwise operator. These operators are undefined for use with **enum** types. The operation is performed; however, the compiler treats the **enum** object as an integer.

User Action: Cast the **enum** object to **int**.

EXTRACOMMA, Extraneous comma in macro parameter list ignored.

Warning: The **#define** macro definition on this line had extra commas that were ignored.

User Action: Make sure that you do not omit parameters in the macro definition.

EXTRAFORMALS, Extraneous formal parameter(s) ignored in declaration of "****"

Warning: You included a function's formal parameters in a function declaration or definition.

For example, the following function declaration is not allowed because it names the function's parameters:

```
int funct(a,b,c);
```

The parameters a, b, and c are ignored.

Similarly, the following example defines a function returning a pointer to a function returning an integer. The names of the parameters of the function returning an integer are not allowed.

```
(*f(p1,p2))(q1,q2)
int p1, p2;
{ . . . }
```

The compiler ignores the parameters q1 and q2.

User Action: Check the syntax of the function declaration and, if appropriate, remove the extraneous identifier(s).

EXTRATEXT, Extraneous text in preprocessor directive ignored.

Informational: Extraneous text appeared in the directive. For example:

```
#endif ABC
```

The compiler issues this message if you specify the **–V standard=portable** option on the **vcc** command line.

User Action: Either remove the extraneous text or enclose it in a comment.

FATALSYNTAX, Fatal syntax error.

Fatal: The compiler could not continue due to syntax errors.

User Action: Correct the error in the indicated line and any errors reported in previous compiler messages.

FILENOTFOUND, Include file could not be opened.

Fatal: The compiler could not find the include file in any of the valid text libraries or directories.

User Action: Check to see if the file exists. See if the include method you used for this file will search for the file in the place you expect.

FLOATOVERFLOW, Overflow during evaluation of floating-point constant expression.

Error: Overflow occurred during the evaluation of a constant expression containing floating-point operands.

User Action: Make sure that the expression value is in the range $0.29 \times 10\_{38}$ to $1.7 \times 10_{38}$.

FUNCNOTDEF, Static function "****" is not defined in this compilation; assumed to be external.

Warning: The indicated static function declaration did not see an existing definition. The compiler treated the function as external.

User Action: Remove the storage-class specifier **static** in the function declaration or use the specifier in the appropriate function definition.

GLOBALENUM, Enumerators may not be initialized when declared with "globalref".

Warning: You tried to specify the values of enumeration constants in a declaration of an **enum** variable with the **globalref** storage-class specifier.

You must define these values elsewhere, in a **globaldef** declaration, and you must not initialize them in the **globalref** declaration.

User Action: Remove all initializing values from the **globalref** declaration.

IFEVALERROR, **** while evaluating #if or #elif expression; "true" expression assumed.

Warning: The substitute text is Stack overflow or Divide by 0.

User Action: For stack overflow, reduce the complexity of the expression. Make sure that no divisors are 0.

IFSYNTAX, Syntax error in #if or #elif expression; true expression assumed.

Warning: The **#if** or **#elif** expression on the indicated line cannot be evaluated because of syntax errors; it was assumed to be true.

User Action: Correct the line.

IGNORED, Unexpected **** ignored.

Warning: The compiler encountered an unexpected token in the source program, and has ignored it. (This may be a syntax error.)

User Action: Make sure that the token and surrounding text is syntactically correct.

INCBUILTARG, Incorrect type for *** argument of "***" builtin function.

Error: An argument to a built-in function has the wrong type.

User Action: Correct the call to the built-in function to pass the correct number and type of arguments.

INLINCONF, Previous inline or noinline pragma for "****" conflicts with this pragma.

Warning: You used both an inline pragma and a noinline pragma specifying conflicting inline specifications for one particular function.

User Action: Determine whether you want the function to be expanded inline, and remove the conflicting pragma.

INSBEFORE, Inserted **** before ****.

Warning: The compiler tried to recover from a syntax error by inserting a token into the source.

User Action: Correct the syntax.

INSMATCH, Inserted **** to match **** on line ****.

INSMATCH, Inserted **** to match **** inserted earlier.

Warning: The compiler tried to recover from a syntax error by inserting a macro to match a previous macro in the source code. The previous macro may or may not have been inserted by the compiler.

User Action: Make sure that you match all parentheses, brackets, and braces.

INTVALERROR, Integer value not used where required.

Error: You used a noninteger value as an initializer for an **enum** constant, or to specify the size of a bit field.

User Action: You must specify an integer constant.

INTVALREQ, Noninteger value used incorrectly in a **** ; converted to integer.

Warning: You used a noninteger value in a **switch** statement or a **case** label. The value has been converted to integer.

User Action: Specify **switch** expressions and **case** label values as integer values, or use a cast operator to make the conversion explicit.

INVAGGASSIGN, Invalid aggregate assignment.

Error: You tried to assign an array to another array or to assign structures or unions of different sizes.

User Action: Correct the assignment.

INVALIDIF, "****" is not a valid constant or operator in a #if or #elif expression; "true" expression assumed.

Warning: You used an invalid construction in an **#if** or **#elif** expression, which is assumed to be true.

User Action: Correct the expression.

INVALIGNSPEC, Invalid alignment specification ignored.

Warning: You specified an alignment option that was not in the range allowed. The compiler ignored the specified option.

User Action: Correct the alignment specification.

INVALINIT, The initialization of "****" is not valid.

Warning: The indicated object cannot be initialized as specified. Some objects may not be initialized at all, such as functions, unions, and **extern** or **globalref** objects. In other cases, the initializer may not be appropriate; for example, a static pointer cannot be initialized with the address of an automatic variable. This and any subsequent initializers for the same object have been ignored.

User Action: Eliminate or correct the initializer, or correct the type or storage class of the target object, or initialize the object with an explicit assignment.

INVARRAYBOUND, The declaration of "****" specifies a missing or invalid array bound.

Error: In a declaration of an array, you omitted a required dimension bound value or specified an invalid value for a bound.

For multidimensional arrays, you must specify bounds for dimensions other than the first. You also must specify a bound for the first (or only) dimension if this declaration is a definition. Valid bound values are integer constant expressions greater than 0.

User Action: Make sure that all required bounds are present and valid.

INVARRAYDECL, "****" is an improperly declared array.

Error: You improperly declared an array, such as an array of functions.

User Action: Make sure that the syntax of the declarator correctly describes the object. (The declared object may not be what you want.) You may find the output from the **-V**"SHOW=SYMBOLS" option on the **vcc** command line helpful to diagnose this error.

INVASSIGNTARG, Invalid target for assignment.

Error: You specified, as the left operand of an assignment operator, an expression that was not valid for assignment. For example, you may have tried to assign something to an array, to a function, to a constant, or to a variable declared with the **readonly** storage-class modifier.

User Action: Make sure that the target is appropriate for assignments.

INVBREAK, Invalid use of the "break" statement.

Error: You used **break** outside the body of a **for**, a **while**, a **do**, or a **switch** statement.

User Action: Remove the **break** statement, or check that any braces in recent loops or **switch** statements are properly balanced.

INVBUILTIN, The "***" builtin function call is being ignored; it has invalid argument(s).

Error: A call to a built-in function contains errors. This message usually follows other error messages describing errors in the argument expressions.

User Action: Correct any errors listed before this one. Make sure that the function is called with the correct number and types of arguments.

INSMATCH, Inserted **** to match **** on line ****.

INSMATCH, Inserted **** to match **** inserted earlier.

Warning: The compiler tried to recover from a syntax error by inserting a macro to match a previous macro in the source code. The previous macro may or may not have been inserted by the compiler.

User Action: Make sure that you match all parentheses, brackets, and braces.

INVCMDVAL, "****" is an invalid command qualifier value.

Fatal: The indicated CC command option value was acceptable to the command language interpreter, but it is meaningless to VAX C; for example, LIST_OPTS is an invalid value for /SHOW, but it is accepted by the **vcc** command.

User Action: Correct the qualifier value.

INVCONDEXPR, The second and third operands of a conditional expression cannot be converted to a common type.

Error: You specified an invalid combination of operands in a conditional expression.

This can occur if the operands are pointers to objects of a different size or type, or if the operands are different structures.

User Action: Make sure that both operands are of compatible sizes and data types.

INVCONST, "****" is an invalid numeric constant.

Warning: The indicated constant contained illegal characters or was otherwise invalid.

User Action: Correct the constant.

INVCONTINUE, Invalid use of the "continue" statement.

Error: You used the **continue** statement outside the body of a **for**, a **while**, or a **do** statement.

User Action: Remove the **continue** statement, or check that any braces in recent loops are properly balanced.

INVCONVERT, The source or target of a conversion is noncomputational.

Error: One of the operands in an expression could not be converted as specified. For example, you tried to cast some object to a structure.

User Action: Correct the expression or cast.

INVDEFNAME, Missing or invalid name in **** preprocessor directive; directive ignored.

Warning: The indicated directive was missing a required name, as in:

```
#define
```

The entire directive was ignored.

User Action: Correct or remove the directive.

INVDICTPATH, Missing or invalid path name in #dictionary preprocessor directive; directive ignored.

Warning: The indicated directive was missing a required name. For example:

```
#dictionary
```

The compiler ignores the entire directive.

User Action: Correct or remove the directive.

INVFIELDSIZE, The declaration of "****" specifies an invalid field size; size of 32 bits assumed.

Warning: The indicated field declaration was invalid because it specified too large a size.

User Action: Correct the declaration to specify either a single, smaller field or several contiguous fields.

INVFIELDTYPE, The declaration of "****" specifies an invalid data type; type "unsigned" assumed.

Warning: You declared a field with an invalid data type. Fields must be declared (and manipulated) as integers or enumerated types.

User Action: Correct the declaration to specify a valid data type.

INVFILESPEC, Missing or invalid file specification in #include preprocessor directive; directive ignored.

Warning: The **#include** directive either was missing a file or specified one that was syntactically invalid. The directive was ignored.

User Action: Correct the directive.

INVFUNCDECL, "****" is an improperly declared function.

Error: You improperly declared a function. For example, you may have omitted the parameter list or a semicolon between the function and a previous declaration.

User Action: Correct the syntax of the declaration.

INVFUNCOPTION, Invalid function definition option "****" ignored.

Warning: The indicated function definition option was not supported. (The only valid option is the main_program option.)

User Action: Check the spelling of the option or the syntax of the function definition.

INVHEXCHAR, Invalid hexadecimal character value; high-order bits truncated.

Warning: An escape character specified in hexadecimal exceeded the limit of a 1-byte character.

User Action: Correct the hexadecimal constant to represent a valid escape character.

INVHEXCON, Hexadecimal constant contains an invalid character.

Error: You specified an invalid hexadecimal constant, such as 0xG.

User Action: Correct the constant.

INVIFNAME, Missing or invalid name in #ifdef or #ifndef preprocessor directive; "true" assumed.

Warning: You specified no name or a syntactically invalid one in the directive; the result of the test is assumed to be true.

User Action: Correct the directive.

INVINAGGASN, Invalid "***" built-in function call; structure or union arguments are not of same size.

Error: A built-in function that requires two or more arguments be of the same size was called with arguments of different sizes.

User Action: Correct the call to the built-in function to pass the correct number and type of arguments.

INVLINEFILE, Invalid file specification in #line preprocessor directive; directive ignored.

Warning: The file specification was syntactically invalid, and the directive was ignored.

User Action: Correct the directive.

INVMAINRETVAL, Return value of main function is not an integer type.

Warning: You declared a main function with a return value that was not an integer type.

User Action: Check for an omitted semicolon at the end of any declaration immediately preceding the declaration of the main function, or change the return value specification to one of the integer types.

INVLINELINE, Missing or invalid line number in #line preprocessor directive; directive ignored.

Warning: The line number was missing or was syntactically invalid, and the directive was ignored.

User Action: Correct the directive.

INVMODIFIER, "****" is an invalid data type modifier in this declaration.

Warning: You specified a data-type modifier other than **const** or **volatile** as in the following example:

```
char * int ptr;
```

The **int** data-type modifier is ignored.

User Action: Remove or change the data-type modifier.

INVOCTALCHAR, Invalid octal character value; high-order bits truncated.

Warning: The octal value in an escape sequence was too large, as in '\477'. Its high-order bits were truncated.

User Action: Correct the value.

INVOPERAND, Invalid **** operand of a "****" operator.

Error: You specified an invalid operand for the indicated operator.

This message is issued for arithmetic and bitwise operators if the operand is noncomputational (such as a structure). For other operators (such as the increment operator), the compiler issues the message if the operand is not an lvalue. For binary operators, the substituted text indicates which operand, left or right, is invalid.

User Action: Make sure that the operand is the proper type for the operator, and that it is an lvalue.

INVPPKEYWORD, Missing or invalid keyword in preprocessor directive; directive ignored.

Warning: You wrote a directive with no keyword. For example:

```
# ABC
```

The directive is ignored.

User Action: Correct or remove the directive.

INVPROTODEF, The parameter list for a function prototype definition must associate an identifier with each type.

Error: The function definition uses the prototype format but does not contain an identifier for each type in the parameter list.

User Action: Place an identifier name in the appropriate type declaration.

INVPTRMATH, Invalid pointer arithmetic.

Error: You tried to perform an invalid arithmetic operation on a pointer or pointers. The only valid arithmetic operations allowed with pointers are addition and subtraction.

For addition, the only forms allowed are as follows:

```
pointer  +  integer
pointer +=  integer
```

For subtraction, the only forms allowed are as follows:

```
pointer  -  integer
pointer -=  integer
pointer  -  pointer
```

In the last form, both pointers must point to objects of the same size.

User Action: Make sure that the expression conforms to one of the permitted forms previously listed. If necessary, cast one or both operands to a compatible type.

INVSUBUSE, Invalid use of subscripting.

Error: You specified a subscript in reference to a bit field.

User Action: Correct the syntax. If the structure containing the bit field is an array, you must specify the subscript(s) with the qualifier instead of the member name.

INVSUBVALUE, Invalid subscript value.

Error: You specified a subscript value that is not of an integer type.

User Action: Change or cast the value to an integer type.

INVTAGUSE, Invalid use of tag "****".

Error: You used a previously defined tag name in a declaration of a different type. For example:

```
enum    color  {red, green, blue};
struct  color  *cp;
```

You may only use a given tag with one of the **enum**, **struct**, or **union** types. Any identifiers declared with the mismatched type will be undefined.

User Action: Either make sure that each use of the tag name specifies the same type or use different tag names with each type.

INVVARIANT, Invalid declaration of variant aggregate "****".

Error: You tried an invalid variant structure or union declaration such as an array of variants, a pointer to a variant, or a list of variant names.

User Action: Either remove the variant keywords from the declaration or make sure that the keywords are used in a valid structure or union declaration.

INVVOIDUSE, "void" is only valid in a parameter list when it appears alone. Its use is ignored.

Warning: You used the **void** keyword in a function prototype parameter list where it is not the only item in the list.

User Action: Either eliminate the use of **void** or eliminate the extra parameter types in the parameter list.

LISTTOOLONG, List in #pragma preprocessor directive is too long; directive ignored.

Warning: You have specified more than 128 items in the list. The entire directive was ignored by the compiler.

User Action: Split the list into separate directives.

MACDEFINREF, A macro cannot be **** during the scan of a reference to the macro; directive ignored.

Warning: You tried to redefine or undefine a macro within a reference to it. The compiler ignores the preprocessor directive.

User Action: Move the directive to a position outside of the macro reference.

MACNONTERMCHAR, Nonterminated character constant in macro argument; apostrophe added at end of line.

Warning: You omitted the closing apostrophe in a character constant appearing in an argument in a macro reference.

User Action: Correct the constant.

MACREQARGS, Macro reference requires an argument list; "****" not substituted.

Error: You wrote a macro reference without an argument list. The reference was deleted from the source file.

User Action: Correct the reference, specifying the same number of arguments as in the definition of the macro.

MACSYNTAX, Syntax error in macro definition; directive ignored.

Warning: The syntax of the parameter list in a macro definition was invalid. (You must enclose the parameter list in parentheses and delimit individual parameters with commas.)

User Action: Correct the syntax.

MACUNEXPEOF, Unexpected end-of-file encountered in a macro reference; "****" not substituted.

Error: The end-of-file was encountered during a macro reference; the reference was deleted.

User Action: See if you misplaced the closing parenthesis in the macro argument list.

MAXMACNEST, Maximum text replacement nesting level exceeded; "****" not substituted.

Error: You specified a macro reference that is recursive or otherwise causes repeated substitutions to a depth greater than the implementation maximum of 64.

User Action: Correct the recursion or simplify the definitions.

MERGED, Merged **** and **** to form ****.

Warning: The compiler merged two separate source tokens into a single token.

For example, two plus signs separated by a space may be merged to form the increment operator (++).

User Action: If the compiler's action is correct, remove the space between the tokens. Otherwise, check for a missing token between the merged tokens.

MISARGNUMBER, The number of arguments passed to the function does not match the number declared in a previous function prototype.

Warning: The function call contains too few or extra arguments.

User Action: Correct the number of arguments passed to the function. Otherwise, if the prototype is incorrect, correct the prototype.

MISPARAMNUMBER, The number of parameters declared does not match the number declared in a previous function prototype.

Warning: A function prototype for this function that appeared earlier in the source file contains a different number of parameters than this declaration.

User Action: Determine which of the declarators is correct and modify the other declarator to match it.

MISPARAMTYPE, The type of parameter "****" does not match the type declared in a previous function prototype.

Warning: The type of a parameter in a function definition does not match the type specified for that parameter in the previous prototype.

User Action: Determine which type is correct for that parameter and correct either the function definition or the prototype.

MISPARENS, Mismatched parentheses in #if or #elif expression; "true" expression assumed.

Warning: The expression in a **#if** or **#elif** preprocessor directive contained unbalanced parentheses.

User Action: Make sure that you balance the parentheses in the expression.

MISPRAGMASTAND, Mismatched #pragma standard preprocessor directive(s)

Informational: The compiler detected more occurrences of the **nostandard** pragma than it did the **standard** pragma.

User Action: Check that each **nostandard** pragma has a matching **standard** pragma, both in the main source file and in any included files.

MISSENDIF, Missing #endif preprocessor directive(s).

Error: The compiler did not encounter an **#endif** line for the most recent **#if**, **#ifdef**, or **#ifndef** preprocessor directive.

User Action: Make sure that all the directives are properly structured, and, if appropriate, add the missing **#endif** preprocessor directive(s).

MISSEXP, Missing or invalid exponent in float constant; zero exponent ('e0') assumed.

Warning: You wrote a floating-point constant with the letter e or E but with no exponent or an invalid exponent. The exponent was assumed to be 0.

User Action: Correct the constant.

MISSPELLED, Replaced **** with ****.

Warning: You misspelled a reserved word.

User Action: Correct the spelling.

MISWIDETYPE, The prototype for this function does not specify the default widened type for the parameter.

Error: A prototype was declared with a parameter having a type that is, by default, widened with old-style function definitions. For example, a **float** is, by default, sized to a **double** for old-style function definitions. If a prototype is in scope with a size of **float**, then the argument will not have the size that the function expects.

User Action: Correct the declaration in the prototype to specify the larger, widened type. If the type is a **float**, then specify **double**.

MODZERO, Constant expression includes mod 0; the result has been replaced with 0.

Warning: The constant expression had an invalid mod expression, such as 5 % 0. The result was 0.

User Action: Correct the expression (but note that its operands must not be floating-point operands).

NAMETOOLONG, Identifier name exceeds 255 characters; truncated to "****".

Warning: VAX C identifiers are limited to a length of 255 recognized characters.

User Action: Shorten the indicated identifier.

NESTEDCOMMENT, Nested comment encountered.

Informational: The compiler detected an opening comment delimiter (/*) within another comment. (VAX C does not support the nesting of comments; the first ending comment delimiter (*/) encountered ends the comment.)

User Action: Check that you have not misplaced a comment delimiter and accidently turned necessary code into comments.

NOBJECT, No object file produced.

Informational: The compiler did not produce an object file due to conditions reported in previous messages.

User Action: Make the corrections suggested by the other message(s).

NOFLOATOP, The **** operand of a "****" operator has been converted from floating-point to integer.

Warning: The compiler converted the operand to an integer.

The left or right operand of the indicated binary operator, or the operand of the indicated unary operator, cannot be of a **float** or **double** type.

User Action: Change or cast the operand to an integral type.

NOLISTING, No listing file produced.

Informational.: The compiler did not create a listing file (usually due to previously reported errors).

User Action: None.

NOMIXNMATCH, The parameter list of a function can either contain all identifiers or all types, but not both.

Error: The parameter list of a function contains some type specifiers and some identifiers that do not have type specifiers.

User Action: To create a valid function prototype, either eliminate the type specifiers or add type specifiers to the identifiers that are missing them.

NONOCTALDIGIT, Octal escape sequence in a character or string constant terminated by a nonoctal digit.

Warning: There was an 8 or 9 in the second or third position of an octal escape sequence. In this case, the digits preceding the nonoctal digit were evaluated, and the 8 or 9 was considered a separate character. The compiler issues this message if you use the **–V standard=portable** option on the **vcc** command line.

User Action: Make sure that the escape sequence contains only octal digits. If the 8 or 9 is separate from the escape sequence, but must immediately follow it, then pad the escape sequence to three digits using leading zeros.

NONOCTALESC, Escape sequence in a character or string constant starts with a nonoctal digit.

Warning: The first of three digits of an escape sequence was an 8 or 9. In this case, the backslash is ignored, and the 8 or 9 was treated as a character. The compiler issues this message if you use the **–V standard=portable** option on the **vcc** command line.

User Action: Make sure that the compiler resolves the ambiguity correctly.

NONPORTADDR, Taking the address of a constant may not be portable.

Informational: You used an ampersand operator with a constant in the argument list of a function call. (VAX C permits this special case, but other compilers may not.)

User Action: If you do not require portability, no action is necessary. Otherwise, correct the line.

NONPORTARG, Passing a structure by value may not be portable.

Informational: You passed a structure by value in a function call or declared a function parameter as a structure. The compiler issues this message if you specify the **–V standard=portable** option on the **vcc** command line.

User Action: If the program must be portable, pass the structure by reference.

NONPORTCLASS, Storage class "****" is not portable.

Informational: This message was issued against the use of the **globalref**, **globaldef**, **globalvalue**, **readonly**, or **noshare** storage-class specifiers. The compiler issues this message if you specify the **–V standard=portable** option on the **vcc** command line.

User Action: No action is necessary if you do not require compatibility with other C compilers. Otherwise, correct the line.

NONPORTCOMP, Comparison of a pointer with a nonzero integer constant or an integer expression may not be portable.

Informational: You compared a pointer to something besides another pointer or the constant 0. This message is issued if you specified **–V standard=portable** option on the **vcc** command line.

User Action: Change the operands or cast them to the same type.

This usage is not portable and is not recommended. The only portable comparison is a comparison between a pointer variable and 0.

NONPORTCONST, Character constant **** may not be portable.

Warning: VAX C allows up to four characters to be specified in a character constant, but other compilers may not. The compiler issues this message if you specify the **–V standard=portable** option on the **vcc** command line.

User Action: If you do not require portability, no action is necessary.

NONPORTCVT, Conversions between pointers and integers may not be portable.

NONPORTCVT, Conversions between pointers to different types may not be portable.

Informational: You converted a pointer or an address expression to an integer type or to a different pointer type, or an integer expression or a nonzero integer constant to a pointer type. Such usage may not be portable and is not recommended. The only portable assignments are between pointers to objects of the same type or conversion of the integer constant 0 to any pointer type. This message is issued only if you specified **–V standard=portable** on the **vcc** command line.

User Action: Use an explicit cast to perform the conversion.

NONPORTINIT, Automatic initialization for "****" may not be portable.

Informational: You initialized an array or structure of the **auto** storage class. The compiler issues this message if you use the **–V standard=portable** option on the **vcc** command line.

User Action: If you require portability, use separate assignment statement(s) to set the initial value(s).

NONPORTOPTION, The "****" function definition option is not portable.

Informational: The VAX C function definition options are VAX C specific and are not portable. The compiler issues this message if you use the **–V standard=portable** option on the **vcc** command line.

User Action: No action is necessary if you do not require compatibility with other C compilers.

NONPORTPTR, The use of an integer value as a pointer qualifier for "****" may not be portable.

Informational: In a reference to a structure or union member accessed by the right arrow (–>) operator, the qualifying expression to the left of the right arrow should have a pointer value. VAX C allows the use of integer values as well, but such usage is not portable. This message

is issued if you specify the **–V standard=portable** option on the **vcc** command line.

User Action: Either use a true pointer expression as the qualifier or cast the integer expression as an appropriate structure or union pointer.

NONPORTTYPE, Data type "****" is not portable.

Informational: You used one of the VAX C specific data types **variant_struct** or **variant_union**. The compiler issues this message if you specify the **–V standard=portable** option on the **vcc** command line.

User Action: No action is necessary if you do not require program portability.

NONSEQUITUR, "****" is not a member of the specified structure or union.

Informational: In a reference to the indicated member name, you specified a qualifier that does not represent the structure or union that the member belongs to.

The reference is valid, because the member name is unique and the offset can be clearly resolved. This use of member names is maintained only for compatibility with older programs.

User Action: If the qualifier is a pointer, cast it as a pointer to the appropriate structure or union.

NONTERMCHAR, Nonterminated character constant; **** assumed.

Warning: The compiler encountered the end of the source line before the end of a character constant. The compiler assumed the indicated value.

User Action: Correct the constant.

NONTERMNULCHAR, Nonterminated character constant contains no characters; '\0' assumed.

Warning: The compiler detected a single apostrophe (') at the end of the source line.

User Action: Check to see if the apostrophe is extraneous; otherwise, correct the constant.

NONTERMSTRING, Nonterminated string constant; quotes added at end of line.

Warning: The compiler encountered the end of the source line before the end of a character string. The compiler inserted a quotation mark (") at the end of the line.

User Action: Check to see if the string should be continued on the following line; if so, insert a backslash (\ ) at the end of the line. Otherwise, check for the missing quotation mark.

NOOPTIMIZATION, Complex control flow caused optimization to be suppressed for procedure or function "****".

Informational: Optimization was not performed for the indicated function.

User Action: To take advantage of optimization, simplify the control flow within the indicated function.

NOSUBSTITUTION, Macro substitution cannot be performed during the scan of a macro reference; "****" not substituted; directive ignored.

NOSUBSTITUTION, Macro substitution cannot be performed during the scan of a macro reference; "****" not substituted; "true" expression assumed.

Warning: You wrote a complex macro reference that contained a preprocessor directive, which in turn contained another macro reference. For example:

```
macref1 ( arg1,
#include MACREF2
    .
    .
    .
,argn)
```

The substitution of MACREF2 is not performed and the directive containing it is ignored. If the directive is **#if** or **#elif**, the expression is assumed to be true.

User Action: Restructure your code so that the directive is not contained within the macro reference.

NOTFUNCTION, Function-valued expression not found.

Error: You used an expression in the context of a function call, but the expression does not evaluate to a function.

User Action: Make sure that the expression properly evaluates to a function; also make sure that you properly dereference any pointer to a function.

NOTPARAMETER, "****" is not listed in the function's formal parameter list; treated as if declared internally.

Warning: You declared the specified identifier as a function parameter, but the identifier does not appear in the parameter list. For example:

```
f(a) int a,b; { . . . }
```

The identifier b does not appear in function f's formal parameter list. Its declaration is not portable, and is probably a coding error. The compiler treats b as if it were declared inside the function definition; in this case, b becomes an automatic variable.

User Action: Correct the declaration or the parameter list.

NOTPOINTER, Address-valued expression not found.

Error: You used an expression in a context requiring a pointer value, but the expression did not evaluate to an address.

User Action: Make sure that the expression evaluates to a pointer value.

NOTSWITCH, Default labels and case labels are valid only in "switch" statements.

Error: You used **case** or **default** as a label outside the body of a **switch** statement.

User Action: Check for unmatched braces that may have prematurely terminated the most recent **switch** statement.

NOTUNIQUE, "****" is not a unique member name in this context.

Error: You used the same member name in more than one structure or union definition, and then used that member name as an offset from some other structure or union. Since the compiler has no way to determine which member definition to use as an offset, a message is generated.

User Action: To avoid ambiguities, try to make all member names unique.

NULCHARCON, Character constant contains no characters; '\0' assumed.

Warning: You used '' for an ASCII NUL character instead of '\0'.

User Action: Use '\0'.

NULHEXCON, Hexadecimal constant contains no digits; 0X0 assumed.

Warning: You specified a constant such as 0X or 0x.

User Action: Be sure that 0 is a valid value in this context; if so, change the constant to 0x0.

PARAMNOTUSED, Macro parameter "****" is not referenced in the definition.

Warning: A macro definition had more parameters than appeared in its substitution. For example:

```
#define m(a,b,c) a*b
```

User Action: Specify the extra parameter in the substitution or, if it is extra, delete it from the parameter list. (This is a possible programming error.)

PARAMREDECL, This declaration of "****" overrides a formal parameter.

Warning: Your source program contained a redeclaration of one of the function's formal parameters. For example:

```
f(a) { int a; }
```

You cannot reference the parameter from within the function.

User Action: If the declaration is misplaced, move it to a position between the function header and the left brace at the beginning of the function body. Otherwise, rename one of the identifiers.

PARSTKOVRFLW, Parse stack overflow.

Fatal: The source code in your program was too complex, containing statements nested too deeply.

User Action: Simplify the program.

PPUNEXPEOF, Unexpected end-of-file encountered in preprocessor directive; directive ignored.

Warning: The compiler detected the end of the source file while trying to read a continuation of a preprocessor directive.

User Action: Check for nonterminated comments, character strings, and other constructs that can span several lines of code.

PRAGMASYNTAX, Syntax error in #pragma preprocessor directive; directive ignored.

Warning: You have incorrectly coded the directive.

User Action: Correct the error. Check for misspellings or punctuation errors.

PTRFLOATCVT, Operand of pointer addition or subtraction converted from floating-point to integer.

Warning: You combined a pointer operand with a floating-point value. For example:

```
int i,*ip;
.
.
.
i = ip + 2.;
```

User Action: Make sure that pointers are used only with other pointers or with integers; in the previous example and similar situations, remove the decimal point from the literal constant.

QUALNOTLVALUE, The qualifier for "****" is not a valid lvalue.

Error: In a reference to a structure or union member accessed by the period operator (.), the qualifying expression to the left of the period must be an lvalue.

User Action: Correct the qualifying expression.

QUALNOTSTRUCT, The qualifier for "****" is not a structure or union.

Informational: In a reference to a structure or union member, the qualifying expression to the left of the period operator (.) or right-arrow operator (->) did not represent a structure or union.

User Action: Check for spelling errors.

REDEFPROTO, This prototype conflicts with either the function definition or with a function prototype that appears earlier in the file.

Warning: The prototype conflicts with a previous declaration of this function in number or type of arguments or in the return type of the function.

User Action: Determine what attribute does not match and what the correct attribute should be. Correct the invalid definition.

REDUNDANT, The operand of the "&" operator is already an address. The "&" is ignored.

Informational: You specified & in front of an array or function name. The compiler issues this message if you specify the **–V standard=portable** option on the **vcc** command line.

User Action: Make sure that you intend to pass the address of the array or function. If you require portability, remove the redundant & operator.

REGADDR, Taking the address of register variable "****" is not portable and causes its storage class to be changed to auto.

Informational: You used the unary ampersand operator (&) to take the address of a register variable. VAX C changes the storage class of the variable from **register** to **auto**. This allows the address of the variable to be taken. The message is used if you specified the **–V standard=portable** option on the **vcc** command line.

REPABBREV, Replaced abbreviation **** with ****.

Warning: You abbreviated a reserved word.

User Action: Complete the spelling of all reserved words.

REPLACED, Replaced **** with ****.

Warning: The compiler replaced an invalid token with a different token. (Programs that contain syntax errors usually generate this message.)

User Action: Check for incorrect syntax.

REPOVERFLOW, Length of replacement text exceeds maximum buffer capacity; "****" not substituted.

Error: The length of the replacement text for a macro reference or the length of the text plus the rest of the line exceeded the implementation's limit.

User Action: Shorten the replacement text or use multiple substitutions to achieve the desired result.

RESERVED, "****" is a reserved identifier; directive ignored.

Warning: You have specified a reserved identifier name in a **#define** or **#undef** preprocessor directive. Such reserved names may not be redefined or undefined. They are as follows:

- **defined**
- __DATE__
- __FILE__
- __LINE__
- __TIME__

User Action: Choose a different spelling for the identifier.

SEMICOLONADDED, Semicolon added at the end of the previous source line.

Warning: A missing semicolon was added to the line before the line numbered in this message.

User Action: Check the previous line carefully and add the semicolon in the appropriate place.

SUMMARY, Completed with **** errors, **** suppressed warning(s), and **** informational messages.

Informational: This message indicates the number of compiler messages (errors, warnings, and informationals) issued during the compilation process. You can suppress informational and warning messages by using the **–V**"WARNINGS=NOWARNINGS" option to the **vcc** command. (see Chapter 2.)

User Action: Consider the individual messages and recompile if necessary.

SYMTABOVFL, The total number of symbol table pages exceeds the implementation's limit.

Fatal: The program was too complex.

User Action: Simplify the program by reducing the number and size of variables and other names, constants, and so forth.

SYNTAXERROR, **** Found **** when expecting ****.

Error: The syntax error shown prevented the generation of an object file.

User Action: Correct the errors shown.

TBLOVRFLW, Internal table overflow, too many procedures, external symbols (psects), or the program is too complex.

Fatal: Either the source file contains too many functions or expressions, or the compiler has overflowed its virtual address space.

User Action: Reduce the size of the source file by dividing it into smaller, separate files, or change the logic of the program to reduce the number of complicated expressions.

TOOFEWMACARGS, Argument list for macro "****" contains too few arguments; missing arguments assumed to be null.

Warning: You wrote a reference to the indicated macro with fewer arguments than were specified in its definition.

User Action: Make sure that the number of arguments in the macro reference is the same as the number of parameters in the definition.

TOOMANYCHAR, Character constant contains too many characters; truncated to ****.

Warning: The length of a character constant exceeded the implementation limit (four characters). The constant was truncated to the indicated value.

User Action: Reduce the length of the indicated character constant to four or fewer characters.

TOOMANYERR, The total number of errors exceeds the limit of 100.

Fatal: The compiler reported more than 100 error messages in this compilation. The compilation ended at this point.

User Action: Correct the errors reported in previous compiler messages and recompile.

TOOMANYFUNARGS, Function reference specifies too many arguments; excess arguments ignored.

Warning: You called a function with more than 253 arguments. The compiler passed only the first 253 arguments; the compiler ignored the remainder.

User Action: Shorten the argument list.

TOOMANYINITS, The initializer list for "****" specifies too many initializers; excess initializers ignored.

Warning: You specified too many initializers for the indicated variable. (If the indicated item is an array or structure, it may be only partially initialized.)

User Action: Make sure that all braces near the initializer sublists are balanced; if the item being initialized is or contains an array, make sure that you account for all dimensions.

TOOMANYMACARGS, Argument list for macro "****" contains too many arguments; excess arguments ignored.

Warning: You wrote a reference to the indicated macro with more arguments than were specified in its definition.

User Action: Make sure that the number of arguments in the macro reference is the same as the number of parameters in the definition.

TOOMANYMACPARM, Parameter list for macro "****" contains too many parameters; excess parameters ignored.

Warning: The number of macro parameters in a **#define** preprocessor directive exceeded the implementation limit of 64.

User Action: Rewrite the macro definition so that it uses 64 or fewer parameters.

TOOMANYSTR, String constant contains too many characters; truncated.

Warning: You wrote a character-string constant whose length exceeded the implementation's limit of 65,535 characters.

User Action: Shorten the string.

TRUNCFLOAT, Double-precision floating-point constant cannot be converted to single precision; 0.0 assumed.

Warning: You specified a double-precision constant in an expression involving a conversion to single precision, but the constant's value was too small to be represented in single precision.

User Action: Ensure that 0 is a valid value in this context; if necessary, redeclare the conversion target as **double**.

TRUNCSTRINIT, String initializer for "****" contains too many characters to fit; truncated.

Warning: If the variable was a simple one-dimensional array, the initializer was truncated (so that the length of the initializer was array-1) and the null byte was added to the end of the array. If the array is a multidimensional array or an array within a structure, the initializer was truncated to the length of the array and a null byte was not added.

User Action: Treat arrays of characters as strings allowing for the null byte at the end of the array. Consider the special case of multidimensional arrays and arrays within structures, especially when you do not want the null byte to be truncated.

TYPECONFLICT, "****" conflicts with a previous data type in this declaration; previous data type ignored.

Warning: You specified more than one data-type specifier in this declaration, and the indicated specifier conflicted with a previous one.

User Action: Check for a missing semicolon in the previous declaration; otherwise, make sure that all specifiers are compatible.

TYPEINLIST, The type of "****" was specified in the parameter list. This declaration is ignored.

Warning: The function definition uses the prototype format but still contains a declaration of this parameter in the parameter declaration section.

User Action: Eliminate the redundant declaration.

UABORT, Compilation terminated by user.

Fatal: The compilation was terminated by a CTRL/C or **kill** command.

User Action: None.

UNDECLARED, "****" is not declared within the scope of this usage.

Error: You referred to an undeclared variable. (You must declare variables before you use them.)

User Action: Check the spelling of the identifier, or add a declaration for it, if appropriate.

UNDECLARED, "****" is not declared prior to this #pragma preprocessor directive; directive ignored.

Warning: This directive lists an identifier that has not yet been declared. The entire directive has been ignored by the compiler.

User Action: Check the spelling of the identifier or add a declaration for it, if appropriate.

UNDEFIFMAC, "****" is not a currently defined macro; constant zero assumed.

Informational: The identifier of a constant expression in an **#if** or **#elif** preprocessor directive was not currently defined as a macro. The expression was evaluated as if the identifier were a constant 0. This message is only generated if **-V standard=portable** is specified on the **vcc** command line.

User Action: Define the identifier as a macro or remove the reference to it.

UNDEFLABEL, Label "****" is undefined in this function.

Error: You wrote **goto** label-name for an undefined label. The scope of a label name is restricted to the function in which it is used as a label; **goto** statements cannot branch to labels inside other functions.

User Action: Check the spelling of the label name or make other corrections as appropriate.

UNDEFMACRO, "****" is already undefined; directive ignored.

Warning: The specified identifier (in an **#undef** directive) was either never defined or occurred in a previous **#undef**. This message is only generated if **–V standard=portable** is specified on the **vcc** command line.

User Action: Remove the **#undef**, or, if applicable, appropriately add the correct definition of the identifier.

UNDEFSTRUCT, "****" is a structure or union type that is not fully defined at this point in the compilation.

Error: You used a name in the context of a structure or union tag, but the name is either undefined or is not yet fully defined as a tag.

User Action: Check the spelling of the name, and make sure that it is fully defined as a tag before using it.

UNEXPEND, Unexpected end-of-**** encountered in #define preprocessor directive; directive ignored.

Warning: The end of the **#define** directive or the end of the source file was encountered before the definition was complete.

User Action: Check for an incomplete comment within the definition, or check for a missing continuation of the directive.

UNEXPEOF, Unexpected end-of-file encountered in a ****.

Error: The compiler encountered the end of the source file while scanning for the end of a string constant or a comment.

User Action: Make sure that string constants and comments are properly terminated.

UNEXPPDIRX, Unexpected **** preprocessor directive encountered; directive ignored.

Warning: The specified directive occurred out of place and was ignored.

User Action: Check the logic of all directives in the program to be sure that it is valid.

UNKSIZEOF, Operand of sizeof has an unknown size; 0 assumed.

Warning: The operand of a **sizeof** operator was an array whose size was unknown at compile time. A size of 0 was assumed.

User Action: Change the declaration of the array to specify the appropriate dimension bound.

UNRECCHAR, Unrecognized character ignored.

Warning: The line contained either a meaningless character or one that appears out of its proper context, such as a pound sign (#) that was not the first character on a line.

User Action: Move or remove the character.

UNRECPRAGMA, Unrecognized pragma; directive ignored.

Informational: You have specified a **#pragma** preprocessor directive that is not recognized by VAX C.

User Action: Correct the syntactic or semantic error that rendered the directive unrecognizable. Common errors include misspelled parameters and ambiguous abbreviations.

VARNOTMEMBER, A variant aggregate must be a member of a struct or union.

Error: You tried to specify a **variant_struct** or a **variant_union** outside an aggregate declaration.

User Action: If you intend to use the structure or union as declared, and if the structure or union is the outermost aggregate in a group of nested aggregates, replace the variant keywords with **struct** or **union**. If you intend to use the structure or union as a variant aggregate, and if the structure or union is otherwise properly declared, nest the declaration within a valid structure or union declaration. If you used the **variant_struct** or **variant_union** keywords in declarations other than structure or union declarations, remove the variant keywords.

VOIDCALL, A "void" function cannot be invoked in a context where a value is expected.

Error: You coded a call to a function declared as **void**, but the call appeared in a context where a return value was expected.

User Action: Move the function call to a different context, or if the function does return a value, declare it to be **void**.

VOIDEXPR, A "void" expression cannot be used in a context where a value is expected.

Error: You cast an expression to be **void**, but the expression was used in a context where its value was required.

User Action: Remove the cast, or move the expression to a context that requires no return value.

VOIDNOTFUNC, "****" is not declared to be a function; only functions may be declared "void".

Error: You declared an object other than a function to be **void**.

User Action: Check the syntax of the declarator. You may find the output produced by the **-V**"SHOW=SYMBOLS" option on the **vcc** command line to be helpful in diagnosing this problem.

VOIDRETURN, A "return" statement in a "void" function may not specify a value; expression ignored.

Warning: You specified a value in a **return** statement within a function declared as **void**.

User Action: Either remove the return value or redefine the function as returning the appropriate data type.

B.3 Diagnostic Messages from the lk Linker

Errors that occur during the linking of your program are reported by the linker. These messages may result from errors in the user program, (such as references to undefined symbols), from missing pieces needed by the linker to build the executable image, or from errors in the compiler or in the linker itself. The linker also sends informational messages in certain situations.

This section describes the messages generated by the lk linker. For information on ld linker messages, see the *ULTRIX Documentation Set*.

In order of greatest to least severity, the four classes of linker diagnostic messages are as follows:

Code	Description
F	Fatal; the linker cannot continue, so it terminates processing immediately.
E	Error; the linker cannot produce an executable image as output. However, the linker does continue processing input.
W	Warning; the cause of the error should be investigated and, if possible, corrected. The linker continues to process input and will produce an output file.
I	Informational; this message only contains information; no user action is necessary.

The following example shows how linker messages are displayed by stderr:

```
%LK-F-CONFQUAL, You have specified options on the lk command line that
are mutually exclusive.
```

Table B–1 is an alphabetical list of linker diagnostic messages. For each message, the table gives a mnemonic, the class of the message, the message text, and an explanation of the message.

Table B–1: Linker Diagnostic Messages

Mnemonic	Error	Message Text and Meaning
ARZEROSYMS	W	an object library 'pathname' member 'name' with an empty symbol table The named member of the indicated object library has a 0-length symbol table.
BADCCC	F	bad compilation completion code ('n') in module 'module-name' file 'pathname' The compilation completion code in the indicated module is invalid. Recompile the module and relink the program.

(continued on next page)

Table B–1 (Cont.): Linker Diagnostic Messages

Mnemonic	Error	Message Text and Meaning
BADPSC	F	transfer address in unknown psect ('n') in module 'module-name' file 'pathname' The module specified a nonexistent psect for the transfer address. Correct the source code, recompile the module, and relink the program.
COMMDEF	I	'pathname': Definition of common 'name' size 'n' The linker encountered a definition of the named common block in the named file. This message occurs if you specify the **–y** option for that symbol on the **lk** command line.
CONFQUAL	F	conflicting qualifiers You have specified options on the **lk** command line that are mutually exclusive.
CRFERR	W	error encountered in Cross Reference The linker encountered an error while cross-referencing a symbol name. Another message will explain exactly what error was encountered.
DSTINTERR	W	internal coding error "routine_name", symbol 'name' The linker encountered a problem while processing the indicated debugger symbol. Submit an SPR on the linker.
EMPTYFILE	W	no modules found in file 'pathname' The named file contains no object code.
EOMFTL	F	link abort specified in module 'module-name' file 'pathname' The end-of-module record in the indicated module specifies ending the link. Correct the errors in the module and relink the program.
EOMSTK	W	'n' UL items left on Linker internal stack in module 'module-name' file 'file-spec' This message indicates an internal error either in the compiler or in the linker. Submit an SPR either on the compiler that generated the module or on the linker.
ERRORISUE	E	completed with errors The link completed, but errors were detected. Correct the errors and relink the program.
ERRORS	E	compilation errors in module 'module-name' file 'pathname' The indicated module generated errors during compilation. The linker will continue, but it is unlikely that the resulting program will run correctly. Correct the source code errors, recompile, and relink the program.
EXCPSC	F	more than 65535 psects defined in module 'module-name' file 'pathname' The indicated module defines too many psects. Correct the problem and relink the program.

(continued on next page)

Table B–1 (Cont.): Linker Diagnostic Messages

Mnemonic	Error	Message Text and Meaning
EXTDEF	I	'pathname': Definition of external 'entity' 'entity-name' The linker encountered a definition of the named entity in the named file. This message occurs if you specify the **–y** option for that entity on the **lk** command line.
EXTREF	I	'pathname': Reference to external undefined 'name' The linker has just encountered a reference to the indicated name in the named file. This message occurs if you specify the **–y** option for that name on the **lk** command line.
FAOFAIL	W	FAO system service failure The linker encountered an error while calling the Formatted ASCII Output (FAO) system service. Further messages will tell exactly what error was encountered.
FATALERROR	F	fatal error message issued The link completed but a fatal error message was issued.
FILETRACE	I	now processing file: 'pathname' The linker is now processing the indicated file. This message occurs if you specify the **t** option on the **lk** command line.
FORMAT	F	file has illegal format One of the files on the command line is neither an object module nor an object library. Correct the file references and reissue the **fort** command.
GSDTYP	W	illegal GSD record type 'type' in module 'module-name' file 'pathname' The indicated module in the indicated file is corrupt; it contains an illegal Global Symbol Dictionary (GSD) record. Correct the problem and relink the program.
ILLARFOR	F	object library 'pathname' has illegal format The linker detected a format error in the indicated object library. Rebuild the library file.
ILLDEFOFF	F	has a symbol definition offset greater than corresponding segment size The file or member defines a symbol whose offset is outside the program segment in which the symbol is defined. Submit an SPR on the compiler that generated the file or member.
ILLFMLCNT	W	minimum argument count 'n' exceeds maximum ('n') in formal specification of symbol 'name' in module 'module-name' file 'pathname' The object records that describe the indicated symbol specify inconsistent values for the minimum and maximum permissible argument counts for calling the routine. Submit an SPR on the compiler that generated the module.

(continued on next page)

Table B–1 (Cont.): Linker Diagnostic Messages

Mnemonic	Error	Message Text and Meaning
ILLMEMMAG	F	object library 'pathname' has a member with an illegal ar_hdr magic number A member in the indicated object library has a bad magic number field. Rebuild the library file.
ILLNAMELEN	F	'entity' 'name' name length ('n') is illegal—not 1 to 'n' in module 'module-name' file 'pathname' The length of a psect, module, or symbol name is not in the specified range. Correct the length and relink the program.
ILLOBJFOR	F	has illegal format The indicated object file or member is not formatted properly. Rebuild the object file and relink the program.
ILLOBJMAG	F	has an illegal magic number The indicated object file or member has a magic number field that contains a value other than OMAGIC, NMAGIC, or ZMAGIC. Rebuild the object file and relink the program.
ILLODDSEG	F	has an odd length text or data segment The text and data segments of an object file or member must have even lengths. Either the file or member is corrupt or there is a bug in the compiler that generated the file or member. First, correct the file or member and rebuild the program. If the error continues to occur, submit an SPR on the compiler that generated the file or member.
ILLRANTOC	F	ranlib file 'pathname' has an illegal table of contents The indicated library's table of contents is corrupt. Run ranlib on the library and relink the program.
ILLRECLEN	W	illegal record length ('n') in module 'module-name' file 'pathname' The indicated module contains a record that is too long or that is inconsistent with the record type. This can be caused by a corrupt file or by a compiler bug. First, correct the file and rebuild the program. If the error continues to occur, submit an SPR on the compiler that generated the file.
ILLRECTYP	W	illegal record type ('n') in module 'module-name' record 'n' file 'pathname' The indicated object record contains a bad type field. This can be caused by a corrupt file or by a compiler bug. First, correct the file and rebuild the program. If the error continues to occur, submit an SPR on the compiler that generated the file.

(continued on next page)

Table B–1 (Cont.): Linker Diagnostic Messages

Mnemonic	Error	Message Text and Meaning
ILLRELOFF	F	has a relocated field outside the corresponding segment The relocated value for a field places it outside the segment in which it is defined (for example, a text segment symbol whose relocated value places it in the data segment). Correct the source code and relink the program.
ILLRELSYM	F	has a relocation field that points to an illegal symbol A relocation field in the file or member points to a symbol that is not a data definition symbol. Submit an SPR on the compiler that generated the file or member.
ILLSTATEVAL	F	current state out of range in lnk$nxtultrec The linker encountered an internal error. Submit an SPR on the linker.
ILLSTRIDX	F	has an illegal string index in a stab entry The linker encountered a symbol table entry with an illegal string table index. Submit an SPR on the compiler that generated the file or member.
ILLSYMNAM	F	file 'pathname' contains a symbol that exceeds the maximum size The indicated file contains a symbol name longer than 255 characters. Reduce the size of the symbol and then recompile and relink the program.
ILLTIR	W	illegal relocation command ('n') in module 'module-name' record 'n' file 'pathname' The indicated object record contains a bad relocation command. This can be caused by a corrupt file or by a compiler bug. First, correct the file and rebuild the program. If the error continues to occur, submit an SPR on the compiler that generated the file.
ILLVPS	W	illegal position ('n') or size ('n') in STO_VPS command in module 'module-name' file 'pathname' The indicated object record contains a bad STO_VPS command. This can be caused by a corrupt file or by a compiler bug. First, correct the file and rebuild the program. If the error continues to occur, submit an SPR on the compiler that generated the file.
ILLZMAGSEG	F	has an illegal zmagic text or data segment size (not multiple of 1024) The object file or member has a header specifying ZMAGIC format, but the text or data segment size is not a multiple of the page size (1024 bytes). Either the file or member is corrupt or there is a bug in the compiler that generated the file or member. First, correct the file or member and rebuild the program. If the error continues to occur, submit an SPR on the compiler that generated the file or member.

(continued on next page)

Table B–1 (Cont.): Linker Diagnostic Messages

Mnemonic	Error	Message Text and Meaning
INSVIRMEM	E	insufficient virtual memory for 'n' pages for cluster 'name' The linker was unable to allocate enough virtual memory in the resulting program to contain the entire program. Consult the link map (obtained by entering the **lk –M** command) to determine why the linker ran out of virtual memory. Then, correct the problem and relink the program.
INTERNERROR	F	internal linker error 'n', please submit spr The linker encountered an internal error. Submit an SPR on the linker, indicating the error number reported in this message.
INTSTKOV	W	linker internal stack of 'n' overflowed in module 'module-name' file 'pathname' The linker overflowed its internal stack while processing the indicated module. Submit an SPR on the linker.
INTSTKUN	W	linker internal stack of 'n' underflowed in module 'module-name' file 'pathname' The linker underflowed its internal stack while processing the indicated module. Submit an SPR on the linker.
INVDSTREC	W	invalid VAX DEBUG Symbol Table record, type 'n' subtype 'n' The linker encountered the indicated invalid debugger record. Either the object file is corrupt or there is a bug in the compiler that generated the object file. First, correct the file and rebuild the program. If the error continues to occur, submit an SPR on the compiler that generated the file.
LIBNAMLNG	W	library module name 'name' has illegal length ('n') The indicated module name is too long. Shorten the name and then recompile and relink the program.
LIBNOTFND	F	'pathname' not found in /lib, /usr/lib, or /usr/local/lib The indicated library, specified using the **fort –l** command, was not found in any of the expected directories. Either move the file or specify a pathname in the **fort –l** command.
LOCDEF	I	'pathname': definition of 'entity' 'entity-name' The linker encountered a local definition of the named entity in the named file. This message occurs if you specify the **–y** option for that entity on the **lk** command line.
LOCREF	I	'pathname': Reference to undefined 'name' The linker has just encountered a local reference to an undefined symbol in the named file. This message occurs if you specify the **–y** option for that symbol on the **lk** command line.

(continued on next page)

Table B–1 (Cont.): Linker Diagnostic Messages

Mnemonic	Error	Message Text and Meaning
MEMBUG	F	memory (de)allocation bug 'n', %X'n', 'n' The linker encountered an internal error allocating or deallocating dynamic memory. Submit an SPR on the linker.
MEMFUL	E	insufficient virtual address space to complete this link There was insufficient process virtual address space or swap file space to complete the link. Either increase your virtual address space or swap file space, or decrease the size of the program you are trying to link.
MODNAM	F	module 'name' name length is illegal—not 1 to 'n' The length of a module name is not in the specified range. Correct the name length and relink the program.
MULDEF	W	symbol 'name' multiply defined in module 'module-name' file 'pathname' The linker encountered a definition for the named symbol in the indicated module, but the symbol is defined. Correct the source code, so the symbol is defined only once, then relink the program.
MULDEFPSC	W	psect 'name' multiply defined in module 'module-name' file 'pathname' The indicated module defines the named program section more than once. Submit an SPR on the compiler that generated the module.
MULPSC	W	conflicting attributes for psect 'name' in module 'module-name' file 'pathname' The named program section is defined with different attributes in different modules of your program. Correct the source code so that all instances of the same psect have the same attributes, then relink the program.
MULTFR	W	multiply defined transfer address in module 'module-name' file 'pathname' More than one object module specifies a transfer address for the program. Delete all but one transfer address and relink the program.
NOEOM	W	no end of module record found in module 'module-name' file 'pathname' The indicated module does not contain an end-of-module record. This is due to a corrupt file or a bug in the compiler that generated the file. First, correct the file and rebuild the program. If the error continues to occur, submit an SPR on the compiler that generated the file.
NOEPM	W	undefined entry mask of symbol 'name' referenced in module 'module-name' file 'pathname' A .MASK directive in a VAX MACRO object module referenced an undefined symbol. Either define the symbol or delete the reference to it.

(continued on next page)

Table B–1 (Cont.): Linker Diagnostic Messages

Mnemonic	Error	Message Text and Meaning
NOIMGFIL	E	image file not created The linker did not create an output program file. Other error messages explain why it did not create the file.
NOMODS	F	no input modules specified (or found) The linker did not encounter any object modules in any of the files specified on the command line. Correct the command line and reenter the command.
NOPSCTS	F	no psects defined in module 'module-name' file 'path-name' The indicated module does not contain any program sections.
NOTOBJLIB	F	file 'pathname' is not an object library The indicated file is not an object library, but the linker expected it to be one.
NOTpsect	W	relocation base set to other than psect base in module 'module-name' file 'pathname' A "set relocation base" command in the indicated module specified a relocation base other than the base of the psect. Submit an SPR on the compiler that produced the module.
NOTXTENT	I	Entry specified is not defined in text. The entry point name specified in the **fort –e** command is not in the program text section of your program. Instead, it is in the data or uninitialized data section. The program entry point must be in the program text section. Correct the source code and relink the program.
NUDFENVS	W	'n' undefined environment(s): This message reports the undefined environments encountered during the link.
NUDFLSYMS	W	'n' undefined module-local symbol(s): This message reports the undefined module-local symbols encountered during the link.
NUDFSYMS	W	'n' undefined symbol(s): This message reports the undefined global symbols encountered during the link.
OLDTOC	W	ranlib library file 'pathname' table of contents not updated since last modification, re-run ranlib on it The timestamp on the library's table of contents is older than the modification date of the library itself. Run ranlib on the file to update the table of contents.
OVRALI	W	conflicting alignment on overlayed psect 'name' in module 'module-name' file 'pathname' The named psect was defined in multiple modules with different alignments. Correct the psect declarations so that they all specify the same alignment, then relink the program.

(continued on next page)

Table B–1 (Cont.): Linker Diagnostic Messages

Mnemonic	Error	Message Text and Meaning
PSCALI	W	psect 'name' alignment ('n') illegal in module 'module-name' file 'pathname' The module specified an illegal psect alignment. Submit an SPR on the compiler that produced the module.
PSCNXR	W	transfer address is not in executable, relocatable psect in module 'module-name' file 'pathname' The transfer address for a module must be in an executable, relocatable psect. Move the transfer address and relink the program.
RECLNG	W	file 'pathname' has a record of illegal length ('n') The file contains a record of either length 0 or longer than 2048 bytes. Either the file is corrupt or there is a bug in the compiler that generated the file. First, correct the file and rebuild the program. If the error continues to occur, submit an SPR on the compiler that generated the file.
RECTYP	W	file 'pathname' record 'n' is illegal ('n') The file has a record with an illegal type field. Either the file is corrupt or there is a bug in the compiler that generated the file. First, correct the file and rebuild the program. If the error continues to occur, submit an SPR on the compiler that generated the file.
RLZEROTOC	W	ranlib object library 'pathname' has an empty table of contents There are no entries in the object library's table of contents. If there are object files in the library, run ranlib to build the table of contents, then relink the program.
SEQNCE	W	illegal record sequence in module 'module-name' file 'pathname' The indicated module contains an illegal sequence of object records. Submit an SPR on the compiler that generated the file.
STALITUDF	W	Stack of undefined literal 'n' in record 'n' in module 'module-name' file 'file-name' The indicated module contains an object command to push a literal value onto the linker's internal stack, but that literal is not defined. Submit an SPR on the compiler that generated the file.
STRLVL	F	illegal object language structure level ('n') should be 'n' in module 'module-name' file 'pathname' A module header record in the indicated module specifies an invalid object language format. Either the object file is corrupt or there is a bug in the compiler that created the module. First, correct the module and rebuild the program. If the error continues to occur, submit an SPR on the compiler that generated the module.

(continued on next page)

Table B–1 (Cont.): Linker Diagnostic Messages

Mnemonic	Error	Message Text and Meaning
TIRLNG	W	object command data overflows record by 'n' bytes in module 'module-name' record 'n' file 'pathname' There is a bad length field in a Text Information/ Relocation record in the indicated module. Either the object file is corrupt or there is a bug in the compiler that generated the file. First, correct the file and rebuild the program. If the error continues to occur, submit an SPR on the compiler that generated the file.
TIRNYI	W	unimplemented TIR command ('n') encountered in module 'module-name' record 'n' file 'pathname' The indicated module contains a Text Information/ Relocation record that is not yet implemented by the linker. Submit an SPR on the compiler that generated the file.
TRUNC	W	truncation error in psect 'name' offset %X'n' in module 'module-name' file 'pathname' The indicated relocatable reference specified a 1-byte or 1-word relative addressing mode, but the defined address of the symbol is longer than 1 byte or 1 word. Correct the reference by using longword relative addressing.
TRUNCDAT	W	computed value is %X'value1' value written is %X'value2' at location %X'addr' This message accompanies the TRUNC message to give more detailed information about the truncation error. 'value1' is the value that the linker tried to store. 'value2' is the truncated version that the linker was able to store. 'addr' is the virtual address in the executable program where the value is stored.
UDEFPSC	W	attempt to reference undefined psect number 'n' in module 'module-name' file 'pathname' The indicated module references a psect index that it did not define. Submit an SPR on the compiler that generated the object file.
ULTOBJFIL	F	object file 'pathname' There is a problem in the object file named by "path-name." Another message describing the problem will follow.
ULTOBJMEM	F	object library file 'pathname' member 'name' There is a problem in a member of an object library. Another message describing the problem will follow.

(continued on next page)

Table B–1 (Cont.): Linker Diagnostic Messages

Mnemonic	Error	Message Text and Meaning
USEUDFENV	W	undefined environment 'name' referenced OR undefined environment number 'n' referenced in psect 'name' offset %X'n' in module 'module-name' file 'path-name' The linker encountered a reference to an environment that was not defined in any of the object modules. Either define the environment in one of the object modules or remove the reference.
USEUDFLSY	W	undefined module-local symbol 'name' referenced in psect 'name' offset %X'n' in module 'module-name' file 'pathname' The indicated location references a module-local symbol that was not defined. Correct the source code so that either the symbol is defined or the reference is removed.
USEUNDEF	W	undefined symbol 'name' referenced in psect 'name' offset %X'n' in module 'module-name' file 'pathname' The indicated location references an external symbol that was not defined. Correct the source code so that either the symbol is defined or the reference is removed.
WARNISUE	W	completed with warnings The linker finished processing the program, but warning messages were issued.
WRNERS	W	compilation warnings in module 'module-name' file 'pathname' The compilation that produced the indicated module generated warning messages. Depending on the nature of the warnings, you may have to correct the errors that caused the warnings and recompile the module.